A generic software framework for Wang-Landau type algorithms

Augustin Chevallier, Frédéric Cazals

A generic software framework for Wang-Landau type algorithms

Augustin Chevallier [*], Frédéric Cazals[†]

March 22, 2020

Abstract

The Wang-Landau (WL) algorithm is a stochastic algorithm designed to compute densities of states of a physical system. Is has also been recently used to perform challenging numerical integration in high-dimensional spaces. Using WL requires specifying the system handled, the proposal to explore the definition domain, and the measured against which one integrates. Additionally, several design options related to the learning rate must be provided.

This work presents the first generic (C++) implementation providing all such ingredients. The versatility of the framework is illustrated with a variety of problems including the computation of density of states of physical systems and biomolecules, and the computation of high dimensional integrals. Along the way, we that integrating against a Boltzmann like measure to estimate DoS with respect to the Lebesgue measure can be beneficial.

We anticipate that our implementation, available in the Structural Bioinformatics Library (`http://sbl.inria.fr`), will leverage experiments on complex systems and contribute to unravel free energy calculations for (bio-)molecular systems.

[*] `Université Côte d'Azur`, `Inria`, France and University of Lancaster, UK; email: chevallier.augustin@gmail.com
[†] `Université Côte d'Azur`, `Inria`, France; email: frederic.cazals@inria.fr

Contents

1 Introduction

1.1 Sampling with the Wang-Landau algorithm

The Wang-Landau (WL) algorithm is a recently developed stochastic algorithm computing densities of states (DoS) of a physical system. From a physics standpoint, WL computes the density of states associated to an energy. From a mathematical standpoint, WL estimates the push forward measure by an energy function of a density in state space, with a control on the relative error. For typical systems, the DoS spans several orders of magnitude and may be close to zero, whence the shear difficulty. WL is a stochastic approximation method falling in the class of adaptive Monte Carlo as it uses an evolving Markov kernel. At each step, WL estimates s the density of states of the system, which requires two main ingredients: the construction of a Markov kernel biased using the current estimation of the DoS – this requires a so-called *move set* or *proposal*; the update of the DoS – referred to as the *learning rule*. (In the sequel, we distinguish between the proposal as just defined, and the *random walk* i.e. the sequence of points used in the algorithm, as output by the Metropolis-Hasting criterion used in WL.)

This state of affairs prompted a variety of developments: since its inception, the Wang-Landau algorithm has spurred the development of many different variants, and numerous studies have been published about their efficiencies.

The learning rule was extensively studied, comparing the flat histogram criterion with deterministic rules and other proposed rules [1, 2]. It was proposed to smooth or dynamically split the histogram [3, 4] or even replace it with a continuous representation [5]. Other suggested merging the Wang-Landau algorithm with parallel tempering [6, 7]. Finally, a wealth of models were studied, ranging from discrete models [8, 1] to molecules such as met-enkephalin (a pentapeptide) [7, 6, 9, 3, 10].

The physical systems studied triggered the development of various proposals. For example, in molecular studies, various proposals were designed, based on molecular dynamics [6, 3], on internal coordinates (dihedral angles) [9, 10], or variants [11, 4]. In a related vein, WL was also used to perform numerical integration. Multidimensional integral may be approximated by a discrete sum of function values multiplies by the measure of points achieving a given value [12]. (Note that the function value plays the role of the density of states of a physical system.) Such calculations are of special interest to study convergence properties, since exact values (for the whole integral or the density of states) make it possible to scrutinize the convergence properties [13]. In this context, it was observed that bin width introduce another kind of saturation error, which call for a refined treatment of function values [13].

In most if not all the aforementioned studies, the reproducibility of results is jeopardized by the absence of software and/or the lack of implementation details.

1.2 Contributions

The previous review of previous work calls for a customizable implementation of WL, capable of accommodating a variety of systems, the variants of the algorithm, and the proposals. Naturally, the corresponding design should not sacrifice any performance.

Systems. We isolate the pre-requisite namely the configuration space, the density, and the energy function within the so-called physical system. This design accommodates discrete and continuous physical systems, as well as systems derived from numerical integration.

WL and variants. The Wang-Landau algorithm has many variants. The flat histogram rule originally proposed has been enhanced by the 1/t rule. However it has been suggested that the flat histogram criterion is not relevant, and other proposal have been suggested. In addition, several updates rule have been proposed [14]. In this work, we present here a generic framework compatible with all these variants.

Move sets–aka proposals. As outlined in [15], the choice of the underlying proposal q and the associated mixing time for Wang-Landau is a core ingredients for fast convergence. Furthermore, in specific situation, like a discrete state space (e.g. the Ising model) or a state space which is a manifold, the creation of a custom proposal respecting the constrains of these state space is mandatory. In this paper we present a generic framework for proposals, allowing to easily create and combine customs adaptive proposals, with the ability to exploit and/or learn features from the configuration space, reducing the mixing time. In addition, we provide several generic proposals on $\mathbb{R}^n$ that uses geometric information to enhance mixing time [15]. Those proposals are readily available for users and extensive experimental results are provided in [15], including for a small bio-molecule.

This work presents the first generic (C++) implementation providing these features, which is integrated to the SBL Structural Biology Library [16] and `http://sbl.inria.fr`. The versatility of the framework is illustrated with a variety of problems including the computation of density of states of physical systems and biomolecules, and the computation of high dimensional integrals.

2 Mathematical pre-requisites

This section present mathematical pre-requisites to make the paper self-contained, and also provides insights on the role of the measure.

2.1 Density of states and calculation by WL

First, it should be noted the measure on state space only requires to be defined up to a constant (in other words, it is a finite measure and not a probability measure). Therefore in this work, we consider a distribution with density $\pi(x)$ defined on a subset $\mathcal{E} \subset \mathbb{R}^D$. Also consider a partition of $\mathcal{E}$ into so-called strata $\{\mathcal{E}_1, \ldots, \mathcal{E}_d\}$. Denoting λ the Lebesgue measure, our problem is to estimate

$$\theta_i^{\pi *} \stackrel{Def}{=} \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})} = \frac{1}{\pi(\mathcal{E})} \int_{\mathcal{E}_i} \pi(x) \lambda(dx). \tag{1}$$

This problem arises in many areas of science and engineering, two of them being of particular interest in the sequel.

At time t, the WL algorithm computes a sequence of estimates $\theta_i^{\pi}(t)$. Let $J(x)$ be the index of the stratum containing point x Points sampled in a given stratum $\mathcal{E}_i$ by WL at time t follows the probability density

$$\pi_{\theta^{\pi}(t)}(x) = \left(\sum_{i=1}^{d} \frac{\theta_i^{\pi *}}{\theta_i^{\pi}(t)} \right)^{-1} \frac{\pi(x)}{\theta_{J(x)}^{\pi}(t)}. \tag{2}$$

The WL algorithm iteratively construct a sequence $\theta^{\pi}(t) = (\theta_1^{\pi}(t), ..., \theta_d^{\pi}(t))$ of estimates for the unknown vector $\theta^{\pi *} = (\theta_1^{\pi *}, ..., \theta_d^{\pi *})$. To do so, the main steps are (see Algo 1) (i) to sample a point according to a Markov kernel $P_\theta(x_t, .)$ depending on the current estimate θ, (ii) update θ using a learning rate γ, and (iii) evolve γ according to two possible strategies known as the $1/t$ regime and the exponential regime. (This latter option comes from the fact that $\log \gamma \leftarrow (1/2) \log \gamma$.) We note that the exponential regime is governed by a flat histogram, which allows checking that all bins are visited evenly. Denoting $\nu_t(i)$ be the number of samples up to iteration t falling into the i-th stratum $\mathcal{E}_i$. The vector $\{\nu_t(i)\}$ is said to verify the **flat histogram** (FH) criterion provided that, given a constant c:

$$\max_{i=1,\ldots,d} \left| \frac{\nu_t(i)}{t} - 1/d \right| < 1 - c. \tag{3}$$

Algorithm 1 Wang Landau

Set $\theta = (1/d, ..., 1/d)$
Set exponential regime = True
Set $\gamma = \gamma_0$ with $\gamma_0 > 1$
while $t < t_{max}$ **do**
  // Sample according to a Markov kernel $P_\theta(x_t, .)$ depending on θ
  Sample $y \sim q(x_t, .)$
  Compute the Metropolis-Hastings acceptance ratio $\alpha = \frac{\pi(y)}{\pi(x)} \frac{\theta_{J(x)}}{\theta_{J(y)}} \frac{q(y,x)}{q(x,y)}$
  Sample $u \sim \text{Unif}(0, 1)$
  if $u < \alpha$ **then**
    Set $x_{t+1} = y$
  else
    Set $x_{t+1} = x_t$
  // Update the DoS
  Set $\theta_{J(x_{t+1})} = \gamma\ \theta_{J(x_{t+1})}$
  Renormalize the estimated DoS θ
  if Exponential regime **then**
    if Flat histogram **then**
      $\gamma = \sqrt{\gamma}$
    if $\gamma < \exp(\frac{1}{t+1})$ **then**
      Set exponential regime = False
      Set $\gamma = \exp(\frac{1}{t+1})$
  else
    $\gamma = \exp(\frac{1}{t+1})$

2.2 Numerical integration

The Wang-Landau algorithm has been used for numerical integration [12, 17, 18]. In the early works, the bin sizes leaded to a saturation of the error. To prevent the error saturation, estimations of the average energy were added in [18]. In practice, the method proposed by [18] to compute an integral $I = \int_{\mathcal{E}} U(x)\pi(x)dx$ relies on the following observation:

$$I = \sum_i \int_{\mathcal{E}_i} U(x)\pi(x)dx \tag{4}$$

$$= \pi(\mathcal{E}) \sum_i \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})} \int_{\mathcal{E}_i} U(x) \frac{\pi(x)dx}{\pi(\mathcal{E}_i)} \tag{5}$$

$$= \pi(\mathcal{E}) \sum_i \theta_i^* \int_{\mathcal{E}_i} U(x) \frac{\pi(x)dx}{\pi(\mathcal{E}_i)}. \tag{6}$$

Following the points distribution in each stratum given by Eq. (2), the average value in a bin is defined by the integral

$$\frac{1}{\pi(\mathcal{E}_i)} \int_{\mathcal{E}_i} U(x)\pi(x)dx. \tag{7}$$

This integral can be computed during WL runtime by computing the average energy value in a given bin. Therefore if the total volume $\pi(\mathcal{E})$ is known, the integral I can be computed with WL.

It should be noted that our formulation slightly differs from [18] because we consider the general case of a finite measure π on $\mathcal{E}$ while they only consider the Lebesgue measure.

2.3 Incidence of the choice of the measure

For two measure π and μ, we define:

$$\theta_i^{\pi*} = \frac{\int_{\mathcal{E}_i} \pi(x)dx}{\int_{\mathcal{E}} \pi(x)dx} = \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})}$$
$$\theta_i^{\mu*} = \frac{\int_{\mathcal{E}_i} \mu(x)dx}{\int_{\mathcal{E}} \mu(x)dx} = \frac{\mu(\mathcal{E}_i)}{\mu(\mathcal{E})}$$

The choice between π and μ has two direct implications. First, $\theta^{\pi*}$ is different than $\theta^{\mu*}$. Second, as seen from Eq. (2), the density sampled by WL in a given stratum are different.

In certain cases changing the measure on the state can improve the convergence speed. Consider the potential energy $U(x) = \|x\|^2$. In high dimension, because of the concentration of measure, most of the points of each stratum will be on the outer boundary with the Lebesgue measure, making it difficult to go from one stratum to a lower one. Changing the measure π to counter this effect and bias towards the interior boundary might be beneficial. On the other, we will inspect one case for which the choice of measure does not matter.

Density of state: change of measure We describe here a method to deduce the density of state for a measure μ while the Wang-Landau algorithm was used with a measure π on state space. To get from one measure to another, we will need the numerical integration capabilities of the software.

For each $\mathcal{E}_i$, WL for the measure π samples points with respect to the density of Eq. (2) in $\mathcal{E}_i$. Therefore we can compute the following integral for each stratum during the runtime of the Wang-Landau algorithm:

$$I_{\mathcal{E}_i} \overset{Def}{=} \frac{\mu(\mathcal{E}_i)}{\pi(\mathcal{E}_i)} = \int_{\mathcal{E}_i} \frac{\mu(x)}{\pi(x)} \frac{\pi(x)}{\pi(\mathcal{E}_i)} dx. \tag{8}$$

Note that the right hand side of the previous equation yields a simple Monte Carlo approximation of $I_{\mathcal{E}_i}$: since $\pi(x)/\pi(\mathcal{E}_i)$ is the probability distribution of points in $\mathcal{E}_i$ sampled by the Wang-Landau algorithm, $I_{\mathcal{E}_i}$ can be approximated by simply computing the average of $\mu(x)/\pi(x)$ of points in $\mathcal{E}_i$.

We rewrite $\theta_i^{\pi*}$:

$$\theta_i^{\pi*} = \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\pi(\mathcal{E})} \tag{9}$$
$$= \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\sum_j \pi(\mathcal{E}_j)} \tag{10}$$
$$= \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\sum_j \frac{\mu(\mathcal{E}_j)}{I_{\mathcal{E}_j}}} \tag{11}$$

Hence:

$$\theta_i^{\pi*} \sum_j \frac{\mu(\mathcal{E}_j)}{I_{\mathcal{E}_j}} = \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \qquad \forall i \tag{12}$$

and diving by $\mu(\mathcal{E})$,

$$\theta_i^{\pi*} \sum_j \frac{\theta_j^{\mu*}}{I_{\mathcal{E}_j}} = \frac{\theta_i^{\mu*}}{I_{\mathcal{E}_i}} \qquad \forall i \tag{13}$$

Finally, using that $\theta_i^{\mu*} = 1 - \sum_{j \neq i} \theta_j^{\mu*}$ we deduce:

$$\theta_i^{\pi*} \frac{\theta_i^{\mu*}}{I_{\mathcal{E}_i}} + \sum_{j \neq i} \theta_j^{\mu*} \left(\frac{\theta_i^{\pi*}}{I_{\mathcal{E}_j}} + \frac{1}{I_{\mathcal{E}_i}} \right) = \frac{1}{I_{\mathcal{E}_i}} \qquad \forall i. \tag{14}$$

Since $\theta_i^{\pi*}$ and $I_{\mathcal{E}_i}$ are estimated during runtime, the system of equation given in Eq. 14 yields a linear system for the unknowns $\theta_i^{\mu*}$, which can therefore be estimated. In doing so, we note that the condition number of the system matters to assess the stability of the calculation.

2.4 Changing the measure: relevant cases

In the previous section, we established that it is possible to use a different measure than the true target measure of which we wish to estimate the DoS. In this section, we will try to shed light on what type of change of measure can or cannot bring an improvement to the estimation of the DoS.

Constant measure on strata. A natural class of measures are measures constant on each stratum. Such case is equivalent to a discrete *energy* space $\{U_1, ..., U_d\}$ with measures of the form $\pi(x) = f_\pi(U(x))$ and $\mu(x) = f_\mu(U(x))$. This specific setting includes the Boltzmann distribution for discrete state spaces such as the Ising models which are of crucial importance for statistical physics.

In these cases, it turns out that the choice of measure have no impact. In fact, Theorem 1 bellow tells us that given a right choice of initial condition, the algorithm will sample the exact same points for these two measures.

Theorem. 1. *For any $\theta_i^\mu(t_0)$, if $\theta_i^\pi(t_0)$ verifies*

$$\theta_i^\pi(t_0) = \frac{f_\pi(U_i)}{f_\mu(U_i)}\theta_i^\mu(t_0), \forall i \tag{15}$$

then the Wang-Landau algorithm will generate exactly the same points with densities μ and π on state space and for all $t \geq t_0$,

$$\theta_i^\pi(t) = \frac{f_\pi(U_i)}{f_\mu(U_i)}\theta_i^\mu(t), \forall i \tag{16}$$

Proof. We have:

$$\theta_i^{\pi *} = \frac{\int_{\mathcal{E}_i} \pi(x)dx}{\int_{\mathcal{E}} \pi(x)dx} = \frac{f_\pi(U_i)Vol(\mathcal{E}_i)}{\sum_j f_\pi(U_i)Vol(\mathcal{E}_j)}$$
$$\theta_i^{\mu *} = \frac{\int_{\mathcal{E}_i} \mu(x)dx}{\int_{\mathcal{E}} \mu(x)dx} = \frac{f_\mu(U_i)Vol(\mathcal{E}_i)}{\sum_j f_\mu(U_i)Vol(\mathcal{E}_j)}$$

Let $\pi_{\theta^\pi(t_0)}$ the biased density sampled by WL at time t_0 when using π as the measure on state space.

$$\begin{aligned}
\pi_{\theta^\pi(t_0)}(x) &= \left(\sum_{i=1}^d \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{\pi(x)}{\theta_{J(x)}^\pi(t_0)} \\
&= \left(\sum_{i=1}^d \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{f_\pi(U_{J(x)})}{\frac{f_\pi(U_{J(x)})}{f_\mu(U_{J(x)})}\theta_{J(x)}^\mu(t_0)} \\
&= \left(\sum_{i=1}^d \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{f_\mu(U_{J(x)})}{\theta_{J(x)}^\mu(t_0)} \\
&= \left(\sum_{i=1}^d \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1} \frac{f_\mu(U_{J(x)})}{\theta_{J(x)}^\mu(t_0)}
\end{aligned}$$

Where the last line is trivial when considering $\left(\sum_{i=1}^d \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1}$ as the renormalisation factor. Finally we deduce

$$\pi_{\theta^\pi(t_0)}(x) = \left(\sum_{i=1}^d \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1} \frac{\mu(x)}{\theta_{J(x)}^\mu(t_0)} \tag{17}$$

Which is the biased density that would be used by WL at time t_0 when using μ as the measure on state space. Let x_{t_0} be the sampled point. Then the updated estimations using γ as the correction factor are:

$$\begin{aligned}
\theta^{\pi}_{J(x_{t_0})}(t_0+1) &= \theta^{\pi}_{J(x_{t_0})}(t_0) * \gamma \\
\theta^{\mu}_{J(x_{t_0})}(t_0+1) &= \theta^{\mu}_{J(x_{t_0})}(t_0) * \gamma \\
\theta^{\pi}_{i}(t_0+1) &= \theta^{\pi}_{i}(t_0) \qquad \forall i \neq J(x_{t_0}) \\
\theta^{\mu}_{i}(t_0+1) &= \theta^{\mu}_{i}(t_0) \qquad \forall i \neq J(x_{t_0})
\end{aligned}$$

which still respects Eq.16. The final result is obtained by recurrence. □

We also conjecture that for smooth densities, having a bin size close to 0 would yield a similar result.

Non constant measures. To improve convergence, it is then necessary for measures to be significantly non constant inside each stratum. As we argued before, it might be beneficial to change the measure to counter concentration of measure near (outer) boundaries in high dimension. However, the large measure changes leads to high variance in the estimation of the $I_{\mathcal{E}_i}$ and high condition number associated to the linear system 14. Therefore, we expect a trade-off between the connectivity of the Markov chain and the complexity of the estimation of $I_{\mathcal{E}_i}$. Experiments confirm this intuition (see section 3.4).

2.5 Boundary condition

In order to restrict the state space to a subset A, two ways are possible. The first is to design a custom proposal q such that for every x in A, the probability $q(x, A^c)$ of going out of A is 0. The second is to reject samples by evaluating the characteristic function 1_A of A. For complex boundary conditions, the first method is usually impossible, and the second can be computationally expensive. However in some cases it is possible to use a filtered predicate, i.e. to write $1_A = 1_B 1_C$ with 1_B being cheap to evaluate and 1_C expensive, see Example 1 below. Assume most of the boundary is determined by B, in the sense that the probability that P_θ leaves C without leaving B is small compared to the probability of leaving A. A simple strategy is to only evaluate 1_B at every step and evaluate 1_C every k steps, with k of the order of magnitude of the complexity of 1_C. Then if $x_{k(t+1)}$ is not in C, the algorithms rolls back to x_{kt}, and rerun the random walk while checking 1_C for each point to find the exit point.

This strategy calls for two comments. First, it requires going back in time to reproduce the same steps of the random to find the exit point. Hence it requires saving the pseudo-random number generators states in addition of the full state of the algorithm every k steps. Second, this method can introduce a bias in the solution since the random walk is allowed to leave C for $k-1$ point as long as it comes back to A for the k-th point. However, under the previously stated assumption, we conjecture that this bias is small.

Example 1. *A common use case is a filter ensuring that a given molecular conformation remains in a prescribed basin below a potential energy threshold C. Checking the former condition requires quenching i.e. minimizing U, a costly operation. We therefore use the indicator functions $1_A(x) = 1_B(x)1_C(x)$, with $1_B(x) = 1$ iff $U(x) < C$, and $1_C(x) = 1$ iff x is in the prescribed basin.*

2.6 Move sets and adaptivity

Adaptivity for WL. When exploring complex systems, as for example the energy landscapes of biomolecular systems, using move sets adapted to the *topography* of the landscape is beneficial, as *flat* or *steep* regions of the landscape are best explored using *large* and *small* steps respectively. The adaptivity over time of the Markov kernel used depends on the particular application targeted, though. When the problem is to generate diverse conformations regardless of a particular target distributions, continuous adaptive schemes can be used [19, 20]. On the other hand, MCMC algorithms targeting a specific distribution must respect certain adaptivity rules [21]. This duality is discussed in [4], which develops *parallel adaptive Wang–Landau*

(PAWL) based on three adaptation mechanisms: first, a bin-split step when the bin boundaries face a population unbalance; second, the use of multiple chains instead of one; third, a Robbins-Monro update of the standard deviation of the Gaussian move set used. It is noted however, that all ingredients add a level of complexity to this modified algorithm whose theoretical convergence yet has to be established.

We also promote adaptivity by localizing move sets on a per bin basis, exploiting the topography of the landscape, based in particular on the geometry of level set surfaces bounding the strata (Fig. 1). The parameters used to define these move sets – and in particular the cone angles are detailed along with the presentation of the classes (Table 1).

Adaptivity and diffusivity assessment. As shown in [15], we assess our adaptive schemes using the *diffusivity* of the system across strata. To this end, we recall the notions of *aggregated transition matrices* (ATM) and *climbing times* [15].

ATM actually come into two guises: for samples proposed by the move set, and for samples accepted and used by WL:

Definition. 1. *Consider an execution of WL. The* aggregated transition matrix *(ATM) for the proposal q, denoted ATM_q, is the $d \times (d+1)$ row stochastic matrix providing the frequencies of transitions between strata corresponding to the moves proposed by q along the execution. (Nb: column $d+1$ corresponds to moves ending outside the bounded region $\mathcal{E}$.)*

The aggregated transition matrix *for WL, denoted ATM_{WL}, is the $d \times d$ row stochastic matrix providing the frequencies of transitions within WL.*

Samples accepted are also used to measure the time taken to travel across all strata:

Definition. 2. *The* climb *across d strata is defined by two times t_0 and t_1 such that*

- $x_{t_0} \in \mathcal{E}_0$ *and* $x_{t_0-1} \notin \mathcal{E}_0$.
- $x_{t_1} \in \mathcal{E}_{d-1}$ *and* $\forall t \in [t_0, t_1[$, $x_t \notin \mathcal{E}_{d-1}$.

The climbing time *is then* $t_1 - t_0$.

3 Experiments

We show the effectiveness of our software package to accommodating a variety of cases, from physical systems to numerical integration.

3.1 Setup

Software setup. The software design presented is available from the Structural Bioinformatics Library (`http://sbl.inria.fr`). See `https://sbl.inria.fr/doc/Wang_Landau-user-manual.html` and the supplemental section 6.

All experiments were conducted as follows. The energy range is discovered at run time, with new bins created when required. The $1/t$ WL update rule is used. Parameters of proposals are discussed on a per example basis.

Tests. We report three types of experiments. We first illustrate the ability of our setup to handle both discrete and continuous physical systems, reporting results for the Ising model and a small biomolecule (alanine dipeptide). Second, we show the versatility and the efficiency of the implementation by presenting numerical integration results, solving one case left open in [13]. Finally, we provide insights on an issue typically overlooked, namely the choice of the measure used in the calculation/integral. More specifically, we show that improved convergence is obtained upon integrating against a suitable measure (Boltzmann versus Lebesgue).

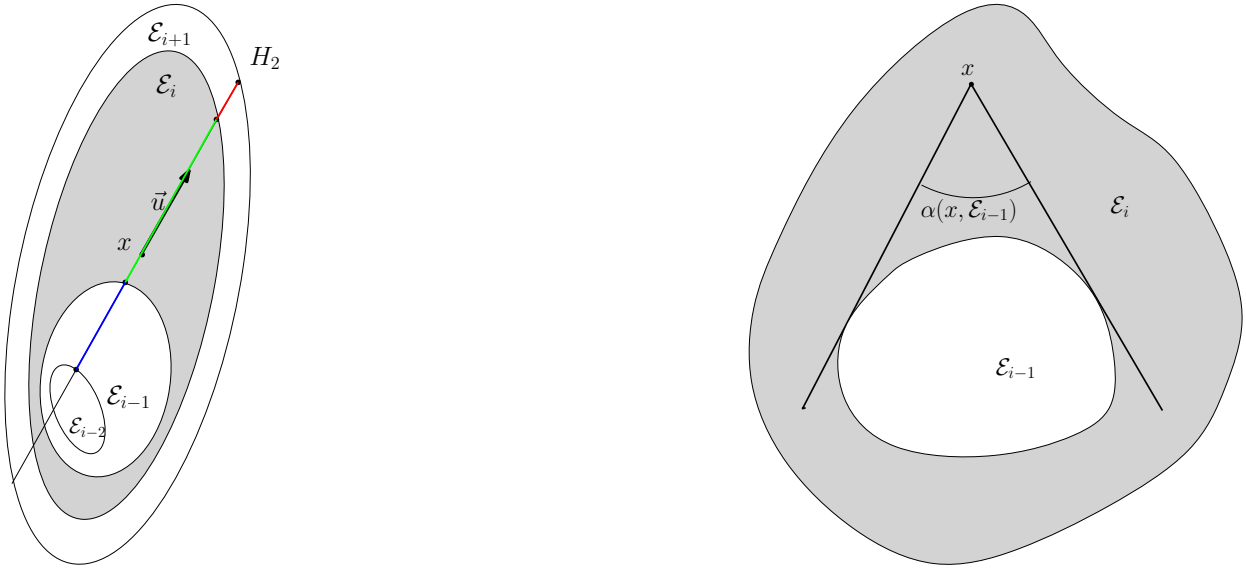

Figure 1: **Adaptive move sets exploiting local features of the energy function. From [15]. (No-overstep move set), class No_Overstep_Move_Set_Traits thereafter** Given a starting point x and a direction u, the no-overstep proposal computes intersections with the level set surfaces bounding the current stratum $\mathcal{E}_i$ as well as the neighboring strata $\mathcal{E}_{i-1}$ and $\mathcal{E}_{i+1}$. These intersection points defined three intervals (blue, green, red). One of these intervals is chosen at random, and a the proposed point is drawn at random from this interval. **(No-overstep cone move set, class No_Overstep_Cone_Move_Set_Traits thereafter)** A cone whose aperture angle is learned is used to enhance the probability of reaching $\mathcal{E}_{i-1}$ from $x \in \mathcal{E}_i$. The random direction chosen in such a cone is then used by the no-overstep move set.

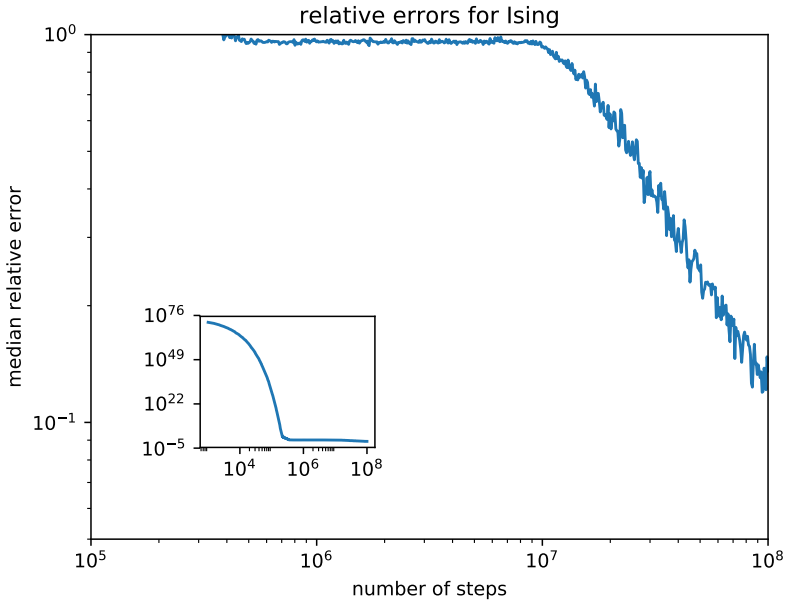

Figure 2: **Relative error for 16x16 Ising model**. Inset: overview for the whole range of relative errors. Median value taken over 40 runs.

3.2 Handling discrete and continuous systems

In the following, we illustrate the ability of our framework to handle discrete and continuous systems, on two classical examples.

3.2.1 Ising model

Ising models are simple yet challenging systems since the number of states is exponential in the system size. Such models were first used to illustrate the strength of the WL algorithm [22]. In the sequel, we consider the 16×16 Ising model, each vertex on the grid having the $+1$ or -1 label. The total number of states is $2^{16\times 16} \sim 1.1510^{77}$. The energy of a given conformation is given by the sum over each vertex of the product of the labels with the neighbors. The proposal flips 1 spin at random in the grid. An analytical formula of the number of configuration for each energy is available [23], which we use to compute the relative error.

The relative error of the estimation computed by the WL algorithm first decreases exponentially, then with the $\frac{1}{\sqrt{t}}$ rate(Fig. 2).

3.2.2 Alanine dipeptide

Dialanine is a small (bio)-system with 22 atoms that is commonly used for benchmarking algorithms. The amber99-sb force field in vacuum is used to compute the density of states between -21 kcal/mol and 4 kcal/-mol, associated to a single local minima of the potential energy function (torsion angles: $\phi = 59.8862, \psi = -35.5193$.) As described in section 2.5, the boundary condition for a local minim requires energy minimisation, which is too expensive to execute every step. Therefore, we use the method described in section 2.5 and minimize the energy only every 100 steps. We compute the following observable akin to the partition function Z, with $\lambda(\mathcal{E})$ the Lebesgue measure of the region of interest of the state space:

$$\frac{Z}{\lambda(\mathcal{E})} = \frac{1}{\lambda(\mathcal{E})} \int_{\mathcal{E}} \exp\left(-U(x)/kT\right) \approx \sum_{\text{Energy levels U}} \theta_i \exp\left(-U/kT\right). \tag{18}$$

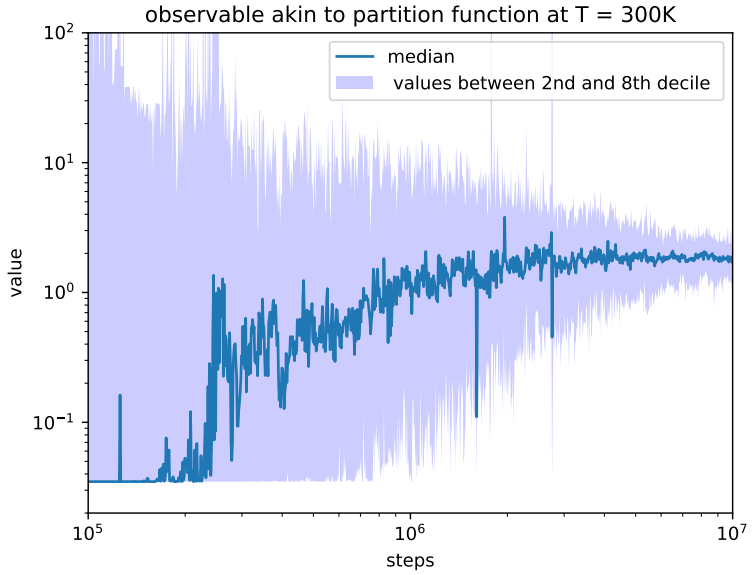

Figure 3: **median and 1rst to 9nth decile of observable akin to the partition function (Eq. 18) for Dialanine**, data obtained for 60 runs.

We performed 60 runs, each with 10^7 steps, and plotted the median of the estimates, as well as the 2nd and 8th deciles (Fig. 3).

3.3 Numerical integration

In cases where direct Monte-Carlo integration is impractical, in particular due to concentration of the measure, numerical integration schemes based on Wang-Landau have been proposed [17, 13]. In the sequel, we study a simple 2D example–as a sanity check, and proceed with one one ill-behaved Gaussian integral.

3.3.1 2D integral

We start with the following test integral from [12, 17, 18]:

$$I_{2D} = \int_{-1}^{1}\int_{-1}^{1} (x_1^6 - x_1x_2^3 + x_1^2x_2 + 2x_1)\sin(4x_1+1)\cos(4x_2)dx_1dx_2 \tag{19}$$

As expected, the decay rate is $1/\sqrt{t}$ (Fig. 4).

3.3.2 Integration of Gaussian density

More interesting is the following integral, studied up to dimension 6 in [13]:

$$I_n = \int f_n(x)dx \text{ with } f_n(x) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{\|x\|^2}{2\sigma^2}}. \tag{20}$$

In the sequel, we use $\sigma^2 = 0.4$. In [13], the integration domain is restricted to $[-10, 10]^n$ to capture most of the mass. However this choice adds artificial difficulties since random walks tend to get stuck in corners of the cube. We therefore change the integration domain to the ball $B(0, 10)$ of center 0 and radius 10. In this setting, I_n is very close to 1 for the dimension range we are interested in.

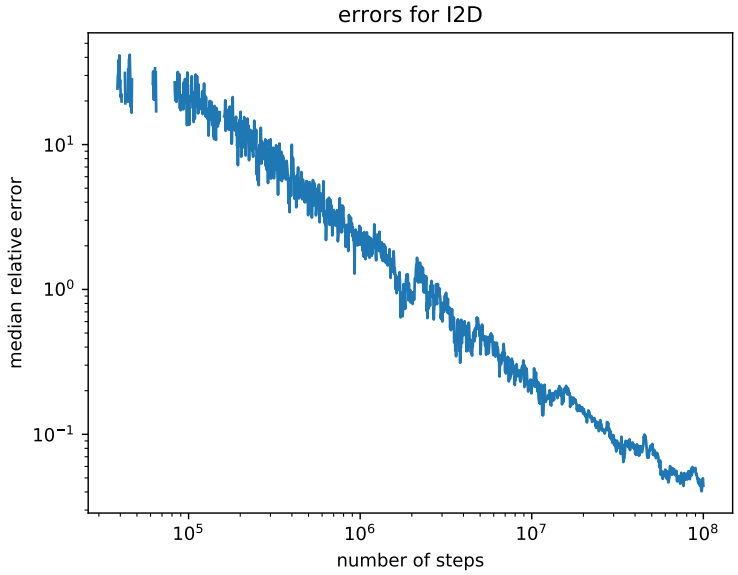

Figure 4: **Median relative error for I_{2D} (Eq. 19)**, data obtained for 40 runs.

This integral is studied up to dimension 6 in [13]. For dimensions 5 and 6 though, the algorithm does not converge, leading the authors to introduce the so-called *two state* strategy which amounts to splitting the integration domain by the median of function values.

In the sequel, we present an effective strategy up to dimension 15 (Fig. 5). When the dimension increases, the volume of the outermost strata increases exponentially; also, since the gradient of the function gets small, so are the components of the gradient, preventing the cone-based proposal to act as an effective guide to diffuse across WL strata, jeopardizing convergence.

To enhance the ability to deal with high dimensions, we introduce a general strategy by taking as energy the logarithm of our integration function:

$$E_n(x) = \log(f_n(x)). \tag{21}$$

This boosts the diffusivity behaviour of the random walk. To get the correct value for the integral, instead of taking the average in each strata, we take the average of the exponential (Eq. 7):

$$I_n(x) = \sum_i V(\mathcal{E}_i) \int_{\mathcal{E}_i} \frac{e^{E_n(x)}}{V(\mathcal{E}_i)} dx. \tag{22}$$

3.4 Incidence of the choice of the measure: experiments

In section 2.3, we have shown how to obtain results for a measure μ on state space while sampling from the base measure π. Practically, this raises the question of choosing the measure to integrate against. we illustrate this by comparing integration results obtained using respectively the Lebesgue and Boltzmann measures. In both cases, we use a Gaussian proposal since cone based proposals (see section 4.3.1) attenuate concentration effects, making the experiment irrelevant.

Consider the following potential $U(x) = \|x\|^2$ for $x \in \mathbb{R}^d (d = 20)$, and an energy discretisation of $[0, 1]$. The aim is to compute the volume of each stratum with respect to the Lebesgue measure. (We make this choice because the exact volume is trivial to compute in the case of the Lebesgue measure.) When the

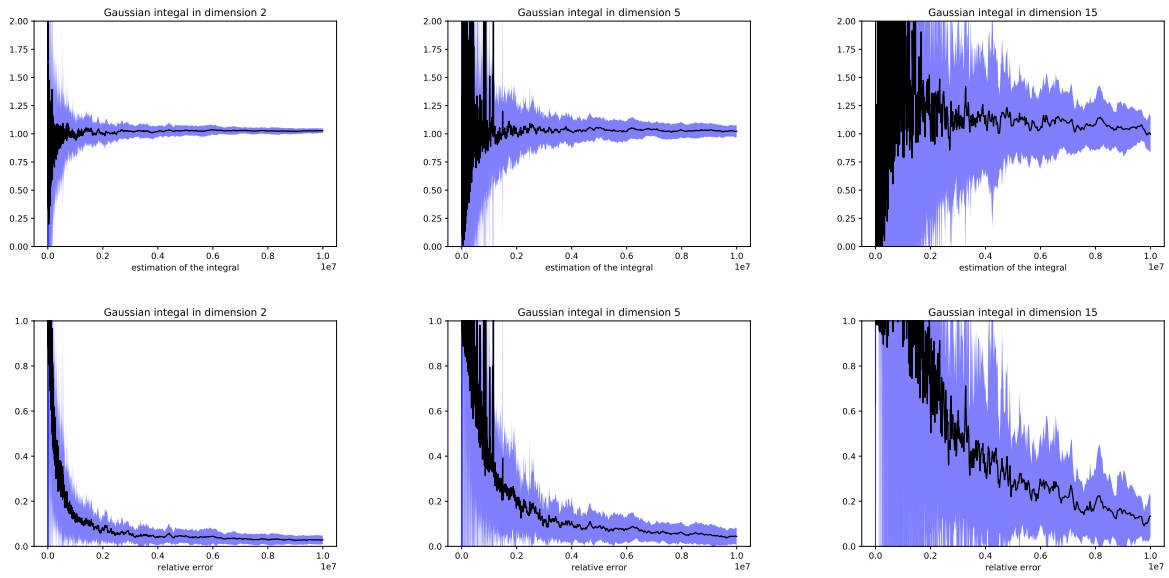

Figure 5: **Value (Top row) and relative error (Bottom row) for the Gaussian integral (Eq. 20) in dimensions 2, 5 and 15, respectively.** A total of 40 runs were performed. The black plot display the median; the purple plot display the 1st and 9th decile.

dimension increases, inside each stratum the volume is concentrated near the outer ring. Hence points sampled by the Wang-Landau algorithm with respect to the Lebesgue measure would be concentrated near this outer ring. This behaviour hinders the convergence speed of Wang-Landau, and a solution consists in splitting the bins if the volume near the outer ring is too imbalanced compared to the interior ring [4].

A case study. We consider a different approach and introduce a Boltzmann-like density

$$\pi_T(x) = e^{-U(x)/T}. \tag{23}$$

Using the Wang-Landau with this density means that points sampled in a stratum are distributed according to π_T, which is larger near the interior sphere than the exterior one. In other words, a suitable temperature T is expected to balance out the measure concentration effects. On the other hand, using $\pi_T(x)$ rather than Lebesgue imposes solving a linear involving the estimations of the integrals of Eq. (8). Poor estimates of these coefficients, or an ill-conditioning of the linear system might yield instabilities.

In the sequel, we study these opposite effects using $T \in [0.01, 10^5]$, this latter value defining an almost flat distribution close to the Lebesgue measure.

We first notice that the best relative error is obtained with an intermediate temperature, namely $T = 0.05$ (Fig.6(A)). The same plot also shows a noticeable improvement in convergence speed and relative error using the Boltzmann distribution with low temperature, as opposed to a high temperature yielding a distribution which is essentially the same than the Lebesgue measure. On the other hand, both the climbing times (Fig.6(B)) and the ATM matrices (Fig.6(C, D)) are best behaved with $T = 0.01$, a value promoting adaptivity. As noticed above, we therefore observe a trade-off between the diffusivity of the proposal and the accuracy of the estimated DoS.

To further this analysis, we analyse two additional statistics. The first one is the condition of the linear system solved. The second one is the average over all bins of the relative standard deviation RSD_i (standard deviation divided by the mean) of $I_{\mathcal{E}_i}$ (computed over 40 runs). It appears that lowering T triggers a dramatic increase of the condition number (Fig.6(E)) and significant increase of the relative standard deviation of $I_{\mathcal{E}_i}$

(Fig.6(F)). As already noticed, these two effects are counterbalanced by the enhanced diffusivity warranted by lower temperatures.

Remark 1. *The quantities involved in the previous linear system may span several orders of magnitude. Practically, we solve this linear system using* `boost::multiprecision::mpfr_float`*, a multiprecision number type.*

4 C++ code design and low-level framework

Our C++ code design is meant to accommodate maximum versatility both in terms of systems processed, move sets used, and inner data structures used by WL. We now review these software components.

4.1 Overview

4.1.1 Components.

The software package is divided into 5 major components (Fig. 7).

Physical system. The *physical* system, provided by the user, specifies the state space and the energy function. It may also provide the gradient of the energy function, if used in conjunction with move set requiring differential information.

Move set and its controller. The move set provides the mechanism to generate a novel conformation given an existing one, and also provides transition probabilities used to ensure detailed balance. The move set works in close collaboration with its so-called controller – see below.

WL algorithm. The *Wang Landau* class is the core one, which glues all pieces together and delivers the DoS estimation. It relies in particular on the *WL data structure*, which stores bin ranges, additional pieces of information detailed below, and provides the bins management policy.

To perform specific processing, like numerical integration, the user must defined a custom WL DS, by inheritance from the default (provided) one.

Helper classes for simulations. We also provide a class *Simulation*, which encapsulates the main ingredients and eases the task of launching a simulation on a given physical system using a given proposal.

4.1.2 Data flow upon generating a point

To present the data exchange between the main components (Fig. 8), we introduce the following parameters and statistics:

- Λ: move set parameters (Class **Move_Params**).
- Ξ: statistics gathered during the generation of a novel conformation y by the move set, and used to tune adaptivity (Class **Move_Stats**).
- Δ: statistics, on a per bin basis, aggregating the observed values of Ξ along the simulation (Class **Controller_Data**).
- θ: DoS estimates.

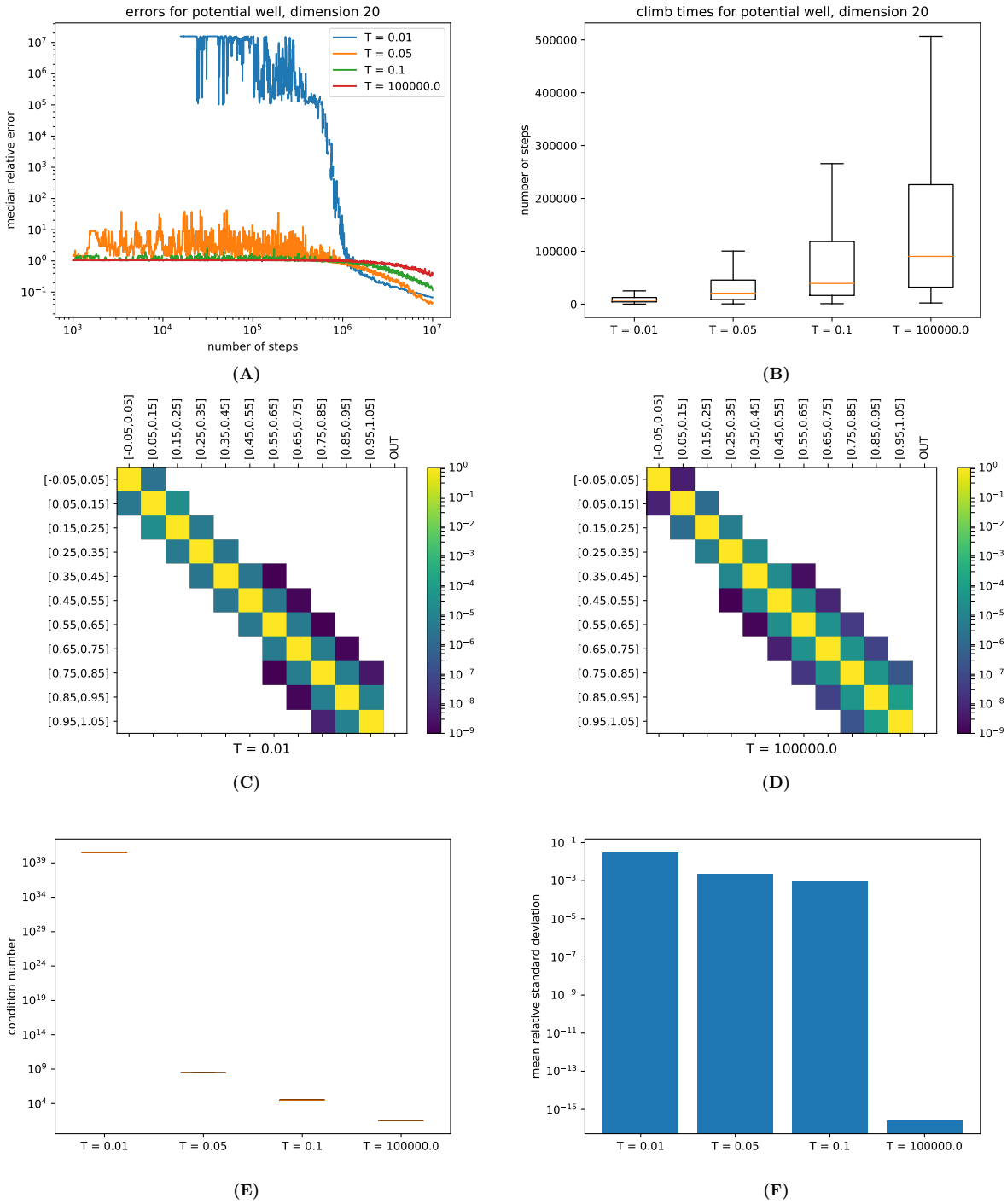

Figure 6: **Computing densities of states: incidence of the choice of the measure.** Experiment for the single well potential in dimension $d = 20$. A Gaussian proposal of variance 0.02 was used; a total of 40 runs were performed. Extreme values of T used are $T = 0.01$ and $T = 10^5$. The DoS may be computed with respect to a density π or μ, see Eq. (9). In this experiment, we use the Boltzmann-like density Eq. 23 to estimate the DoS wrt to the Lebesgue measure. **(A)** Median relative errors with respect to Lebesgue measure as a function of the temperature used in Eq. (23). **(B)** Boxplot of climbing times for the four temperatures – Def. 2. **(C)** $\mathrm{ATM}_{\mathrm{WL}}$ of the first run (Def. 1) for $T = 0.01$. **(D)** $\mathrm{ATM}_{\mathrm{WL}}$ for the first run for $T = 10^5$. **(E)** Boxplot of onditions number for the linear system of Eq. 14, log scale. (The boxplot is flat since all simulations are converged.) **(F)** Relative standard deviation RSD_i of $I_{\mathcal{E}_i}$ averaged over all bins. Log scale.

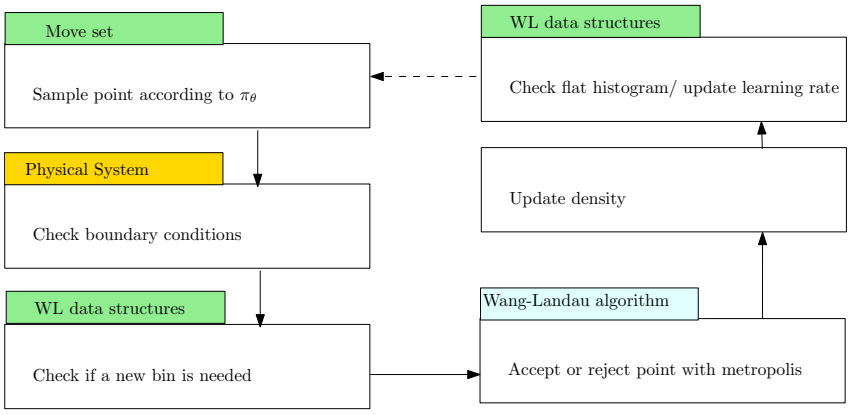

Figure 7: **Algorithm overview: from sample generation to the update of DoS.** Color codes are as follows: gold: Must be provided; green: Can be provided – default exists; cyaun: Must use – provided by the library.

Λ is expected to exist for all move sets. For example, for an isotropic Gaussian move set, Λ is just the kernel width. The quantities Ξ and Δ are meant to provide local information on the energy function, so as to adapt the move set to local features of the energy landscape. (Incidentely, move sets devoid of adaption yield empty Ξ and Δ.) See Table 1.

Using these parameters and statistics, the generation of a novel sample by the move set and its exploitation by WL are as follows:

- Step 1. The controller accesses Δ from the WL DS.
- Step 2. Using Δ for the particular bin containing x, the controller generates the move set parameters Λ, which are passed by WL to the move set.
- Step 3. Using Λ and the current conformation x, the move set generates a candidate conformation y.
- Step 4. From y, the physical systems returns to the move set the energy (and possibly gradient) at y.
- Step 5. WL uses the previous information to update the DoS, and passes Ξ to the move set controller.
- Step 6. Finally, Ξ is used to update the controller data Δ

We now review the main components and these steps in detail.

4.2 Physical system

Classes: User defined

The physical class defines the conformation type, an energy function, the boundary condition and if required, the target density π on state space. If not specified, $\pi(x) \propto 1$, see section 2.3. By *conformation*, we refer to the class storing the point in state space and the corresponding energy $U(x)$. For proposals requiring the gradient, the conformation also stores $\nabla U(x)$.

In practice, for continuous systems, conformations are stored as a vector of doubles. For systems whose state space is discrete, conformations are typically stored as an integer.

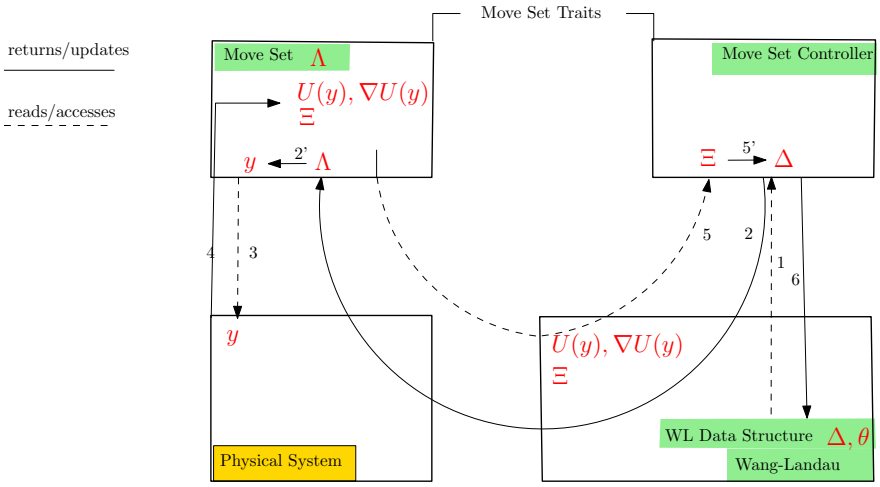

Figure 8: **Data flow and update of data structures upon generation of a point by the move set.** Arrows indicate the ordering for data exchange between the main components. See text for details.

4.3 Proposals in the context of a calling algorithm

Proposals play a pivotal role in various stochastic algorithms, including (energy) landscape exploration algorithms in the lineage of basin hopping, and density of state calculations in the lineage of Wang-Landau. We therefore outline a general design for proposals, discussing specific features for WL when applicable.

Strictly speaking, a proposal must provide two operations:

- generate a conformation y given a conformation x and parameters Λ.
- provide the probability $q(x, y \mid \Lambda)$ to move from x to y. This information is used to satisfy detailed balance if required, which is the case for WL.

Upon generating a conformation generation, the move set statistics Ξ are updated.

4.3.1 Elementary proposals provided

Classes: **Gaussian_Move_Set_Traits**,
No_Overstep_Move_Set_Traits, **No_Overstep_Cone_Move_Set_Traits**, **Darting_Move_Set_Traits**

When the conformational space is $\mathbb{R}^n$, the following proposals are provided ([15] and Fig. 1):

- isotropic Gaussian proposal (**Gaussian_Move_Set_Traits**),
- no-overstep (**No_Overstep_Move_Set_Traits**),
- cone based no-overstep (**No_Overstep_Cone_Move_Set_Traits**),
- darting proposal (**Darting_Move_Set_Traits**).

The parameters Ξ and Δ for these proposals are described in Table 1). Consider for example the cone based no-overstep move set: Ξ provides information on the interval hit (Fig. 1(A)); Δ is used to learn the best cone to be used for all points x in a given stratum $\mathcal{E}_i$.

	Λ	Ξ	Δ
Gaussian Move Set	variance σ	$\emptyset$	$\emptyset$
No-overstep Move Set	$\emptyset$	$\emptyset$	$\emptyset$
Darting move set	$\emptyset$	$\emptyset$	$\emptyset$
No-overstep Cone Move Set	• uses cone • cone proba. • cone angle	• interval up • interval down • used cone–if any	Learning phase: • proportion intervals up • proportion intervals down • statistics on candidate cones Exploitation phase: • uses cone (boolean) • cone aperture angle

Table 1: **Description of parameters Λ, Ξ, Δ for the move sets from [15].** See section 4.1.2 for the definition of the data structures. Upon generating a new point, statistics Ξ are used to update those stored on a per bin basis Δ. These parameters are used to update Λ i.e. the parameters used by the move set.

4.3.2 Combined proposal

Classes: **Combined_Move_Set_Traits**

To go beyond unitary move sets, move sets can be recursively combined [15]. The generic pattern to describe such move sets is as follows.

A *combined proposal* is composed of a collection of $k > 0$ proposals $\{MS_1, ..., MS_k\}$, and a function

$$\begin{aligned} w : \mathcal{E} &\longrightarrow \mathbb{R}^k \\ x &\longrightarrow (w_1(x), ..., w_k(x)) \end{aligned}$$

such that for all $x \in \mathcal{E}$, $\sum_l w_l(x) = 1$. For a given $x \in \mathcal{E}$, $w_l(x)$ represents the probability of choosing the proposal MS_l to generate the next point. The parameter for such a proposal is a vector Λ with each coordinates Λ_l being the parameter for the proposal MS_l. Denoting q_l the transition kernel of the proposal MS_l, a combined proposal transition kernel is given by

$$q(x, y \mid \Lambda) = \sum_{l=1}^{k} w_l(x) q_l(x, y \mid \Lambda_l). \tag{24}$$

The generic C++ class (**Combined_Move_Set_Traits**) takes as template arguments traits classes for the move sets embarked.

Remark 2. *(Internal representation of combined move sets) Practically, a combined proposal is represented as a tree. Internal nodes are the combined proposals, and samples are generated by proposals associated with leaves.*

4.3.3 Adaptivity via controller

Classes: **Data_Interface**, **Move_Set_Controller_Data**

The parameters Λ used to define the move sets depend on specific requirements of the algorithm. In our case, the fact that WL uses bins calls for a specialization of Λ on a per bin basis. We guarantee a proper communication between the move set and the WL data structure by means of a *controller*. The controller is a class endowed with a data structure of type **Move_Set_Controller_Data**, denoted Δ, which is used to generate the move set parameters Λ, and which is updated from the move set statistics Ξ. Practically, the calling algorithm (WL in our case) imposes an algorithm dependent data interface (**Data_Interface**) to access Δ.

Remark 3. *(Data structure Δ) In the case of Wang-Landau, the class **Move_Set_Controller_Data** is instantiated once per bin. The move parameters Λ generation by the controller defines a mapping from the space controller data$\Delta \times$ Conformational space $\mathcal{E}$ to Λ, see Eq. (24). Note that one could also directly record data in the controller. However, in storing Δ_i within the WL DS, one automatically benefits from the dynamics (ie creation) of bins.*

Remark 4. *(Controller and adaptivity for combined proposals.) For the combined proposal, the controller will be automatically generated. The data stored (**Move_Set_Controller_Data**) is the reunion of the data used by each controller, and the controllers of the proposals composing the combined proposal are automatically called when relevant. In the WL case, it means that each **Move_Set_Controller_Data** is instantiated once per bin for all the proposals composing the combined proposal.*

4.4 WL data structure

Classes: **T_WL_Data_Structure_MS**

In the sequel, we briefly detail the main functions and customization point of the WL data structure class, denoted **T_WL_Data_Structure_MS**.

4.4.1 Bins data structure

Classes: Bin_properties_MS, Move_Set_Controller_Data

The WL data structure handle bin creation and splitting. The user can specify the behaviour of the data structure when a new bin is required by giving a default bin size and whether the energy levels are continuous or not (e.g., the Ising model has discrete energies). We leave out the implementation details of this behaviour. A type **Bin_properties_MS** defines the default properties to store for each bin: $\theta(i), \nu(i)$ and an instance of **Move_Set_Controller_Data** Δ_i.

Remark 5. *(Storing additional statistics) One must define a class **Custom_Bin_Properties** inhering from **Bin_properties_MS** and pass this type as a template parameter to **T_WL_Data_Structure_MS**.*

4.4.2 Update after point generation

The WL data structure updates relevant information stored in the data structure each time a point is generated by the algorithm. To that end, the WL data structure performs the following operations – see also step 5 in section 4.1.2:

- Step 5a. check the energy $U(x)$ of a generated conformation and possibly create a bin. Reset the learning rate γ upon addition of a new bin.
- Step 5b. check the flat histogram criterion and update the learning rate γ
- Step 5c. update the estimated density i.e. θ if x is accepted

Remark 6. *(Customization.) This default behaviour can be customize to perform the following: use a different learning strategy, possibly introducing variants of the flat histogram rule or record additional statistic. For example, the average energy value per bin (Eq. 7) can be recorded to perform numerical integration, see section 2.2. Practically, a new class inheriting from* ***T_WL_Data_Structure_MS*** *must be provided.*

4.5 Main algorithm

Classes: T_Wang_Landau

The main class (**T_Wang_Landau**) implements the WL algorithm using the previous ingredients (Fig. 7). The corresponding C++ template class requires the following template parameters: the physical system, the proposal (and its controller), the WL DS (default suitable for most cases **T_WL_Data_Structure_MS**).

4.6 Helper classes

Running a simulations. Classes: T_WL_Simulation

A common use case is that where one wishes to compare the performances of various proposals on various physical systems.

To ease such experiments, we provide the class template class (**T_WL_Simulation**, call it A for conciseness) templated by two ingredients: the physical system and the proposal. This class admits two specialized classes, one instantiating the physical system (call it B) and one instantiating the proposal (call it C). From these two classes, one can derive a class D inheriting from classes B and C. This design makes it possible to avoid the duplication of the code inherent to the physical system (found only in class B) and to the proposal (found only in class C). Practically, this multiple inheritance problem is handle using the C++ diamond patter – see e.g. `https://en.wikipedia.org/wiki/Multiple_inheritance`.

Performing a change of measure Classes: WL_histo_change_measure The class performs the change of measure described by section 2.3. It computes θ^μ given the vector of $(I_{\mathcal{E}_i})$ and the result of a Wang-Landau run θ^π by solving eq. 14. Since solving this system usually fails when double precision is used, this class uses `boost::multiprecision::mpfr_float` internally, freeing the user of numerical issues.

4.7 Output

Default pieces of information. The algorithm saves datas contained in the WL Data Structure a given times on a class called T_DS_snapshot. This data structure contains by default:

- Bin ranges (useful if bins are added at runtime)
- Estimated DoS θ^π
- Move set data Δ per bin
- The aggregated transition matrices and climbs/descents times (Section 2.6)
- The climbing and descending times
- The number of times the Flat Histogram criterion was reached
- The algorithm mode (flat Histogram regime or 1/t regime)
- The number of steps since the beginning of the algorithms

These pieces of information can be either saved at fixed intervals or at intervals that scales exponentially with time for log plots.

WL_stats_serializer An helper class is provided to save the previous results in files, that can be later used by (provided) Python scripts to generate figures. This class allows to save: (i) The DoS θ with bin ranges, (ii) The ATM matrices, and (iii) the climbing and descending times. A Python script is provided to plot the ATM matrices from the serialized results. See the jupyter notebook available at `https://sbl.inria.fr/doc/Wang_Landau-user-manual.html`.

5 Outlook

The Wang-Landau (WL) algorithm is a stochastic algorithm initially designed to compute densities of states for physical systems, also suitable to handle challenging numerical integration cases in high dimensional spaces. To the best of our knowledge, this work presents the first versatile implementation of WL, based on a generic C++ architecture encompassing a wide range of WL variants and use cases. On the one hand, end-users can easily perform calculations of discrete and continuous systems, which merely requires plugging the specification of the system into our design. On the other hand, developers can easily test various building blocks and combinations thereof, and select those best suited for a system. The most crucial aspects are the energy discretization strategy and the proposal, which, if poorly chosen, jeopardize convergence. We note in passing that the default implementations provided for proposals are state-of-the-art and should prove useful for future benchmarking. We therefore anticipate that our framework will enjoy applications in physics (partition function), numerical integration, as well as machine learning (computation of maxima a posteriori), on systems involving from tens to hundreds of degrees of freedom.

We foresee developments in several directions, two of which are particularly important. On the methodological side, our framework should prove instrumental in testing novel ideas revolving around the bin structure used in WL. More elaborate adaptation mechanisms for move sets should be investigated, so as in particular to go beyond the adaption on a per bin basis used in this work. Needless to say that these developments will also raise delicate convergence analysis questions. Finally, on the application side, we believe that the ability to use move sets in internal coordinates will make it possible to use WL on molecular systems of intermediate complexity (peptides involving tens of amino-acids), possibly providing valuable thermodynamic estimates for systems of high biological interest.

Acknowledgments. The authors wish to thank Tony Lelièvre for stimulating discussions.

References

[1] A. Swetnam and M. Allen. Improving the wang–landau algorithm for polymers and proteins. *Journal of computational chemistry*, 32(5):816–821, 2011.

[2] B. Werlich, T. Shakirov, M. Taylor, and W. Paul. Stochastic approximation Monte Carlo and wang–landau Monte Carlo applied to a continuum polymer model. *Computer Physics Communications*, 186:65–70, 2015.

[3] Hiromitsu Shimoyama, Haruki Nakamura, and Yasushige Yonezawa. Simple and effective application of the wang–landau method for multicanonical molecular dynamics simulation. *The Journal of Chemical Physics*, 134(2):024109, 2011.

[4] L. Bornn, P. Jacob, P. Del Moral, and A. Doucet. An adaptive interacting Wang–Landau algorithm for automatic density exploration. *Journal of Computational and Graphical Statistics*, 22(3):749–773, 2013.

[5] A. Farris, Y-W. Li, and M. Eisenbach. Histogram-free multicanonical Monte Carlo sampling to calculate the density of states. *Computer Physics Communications*, 235:297–304, 2019.

[6] Nitin Rathore, Thomas A Knotts IV, and Juan J de Pablo. Density of states simulations of proteins. *The Journal of chemical physics*, 118(9):4285–4290, 2003.

[7] C. Junghans and U.H. Hansmann. Numerical comparison of WANG–LANDAU sampling and parallel tempering for met-enkephalin. *International Journal of Modern Physics C*, 17(06):817–824, 2006.

[8] F. Lou and P. Clote. Thermodynamics of RNA structures by wang–landau sampling. *Bioinformatics*, 26(12):i278–i286, 2010.

[9] Pedro Ojeda-May and Martin E Garcia. Electric field-driven disruption of a native β-sheet protein conformation and generation of a helix-structure. *Biophysical journal*, 99(2):595–599, 2010.

[10] P. Poulain, F. Calvo, R. Antoine, M. Broyer, and P. Dugourd. Performances of Wang-Landau algorithms for continuous systems. *Physical Review E*, 73(5):056704, 2006.

[11] Katie A Maerzke, Lili Gai, Peter T Cummings, and Clare McCabe. Incorporating configurational-bias Monte Carlo into the Wang-Landau algorithm for continuous molecular systems. *The Journal of Chemical Physics*, 137(20):204105, 2012.

[12] R. Belardinelli, S. Manzi, and V. Pereyra. Analysis of the convergence of the 1/ t and Wang-Landau algorithms in the calculation of multidimensional integrals. *Physical Review E*, 78(6):067701, 2008.

[13] W. Atisattaponga and P. Marupanthornb. A 1/t algorithm with the density of two states for estimating multidimensional integrals. *Computer Physics Communications*, 220(122–128), 2017.

[14] P. Jacob and R. Ryder. The Wang–Landau algorithm reaches the flat histogram criterion in finite time. *The Annals of Applied Probability*, 24(1):34–53, 2014.

[15] A. Chevallier and F. Cazals. Wang-landau algorithm: an adapted random walk to boost convergence. *J. of Computational Physics*, NA(NA), 2020.

[16] F. Cazals and T. Dreyfus. The Structural Bioinformatics Library: modeling in biomolecular science and beyond. *Bioinformatics*, 7(33):1–8, 2017.

[17] Ying Wai Li, Thomas Wüst, David P Landau, and HQ Lin. Numerical integration using wang–landau sampling. *Computer physics communications*, 177(6):524–529, 2007.

[18] W. Atisattapong and P. Maruphanton. Obviating the bin width effect of the 1/t algorithm for multidimensional numerical integration. *Applied Numerical Mathematics*, 104:133–140, 2016.

[19] L. Jaillet, F.J. Corcho, J-J. Pérez, and J. Cortés. Randomized tree construction algorithm to explore energy landscapes. *Journal of computational chemistry*, 32(16):3464–3474, 2011.

[20] A. Roth, T. Dreyfus, C.H. Robert, and F. Cazals. Hybridizing rapidly growing random trees and basin hopping yields an improved exploration of energy landscapes. *J. Comp. Chem.*, 37(8):739–752, 2016.

[21] J.S. Rosenthal and G.O. Roberts. Coupling and ergodicity of adaptive MCMC. *Journal of Applied Probablity*, 44:458–475, 2007.

[22] F. Wang and D.P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86(10):2050, 2001.

[23] T. Schultz, D. Mattis, and E. Lieb. Two-dimensional ising model as a soluble problem of many fermions. *Reviews of Modern Physics*, 36(3):856, 1964.

6 Appendix: using the Wang-Landau package

In the following, we sketch the steps required to reproduce the results presented in this paper, on a linux like operating system.

6.1 Getting the SBL

To get the SBL, first clone it as follows:

```
git clone git://sbl.inria.fr/git/sbl.git
```

To proceed, set the following environment variables:

- $SBL_DIR: the directory containing the sources of the SBL, obtained with git clone as indicated above.
- $SBL_DIR_INSTALL: the directory into which the executables (binaries) are installed. This directory is expected to be in one's PATH.
- $PYTHONPATH: the python PATH, to which one should add $SBL_DIR/python/SBL and and $SBL_DIR/python. Nb: Python3 must be used.

6.2 Boltzmann energy single well: reproducing the results

To compile the executables:

```
cd $SBL_DIR/Core/Wang_Landau/examples/Wang_Landau/
mkdir build
cd build
cmake .. -DSBL_APPLICATIONS=ON -DCMAKE_INSTALL_PREFIX=${SBL_DIR_INSTALL}
make
make install
```

Once the executable Wang_Landau/examples/Wang_Landau/boltzmann_energy_single_well_figures.exe has been generated, proceed with the jupyter notebook as explained at `https://sbl.inria.fr/doc/Wang_Landau-user-manual.html`:

```
cd $SBL_DIR/Core/Wang_Landau/demos
jupyter notebook Wang_Landau.ipynb
```

6.3 Using the C++ classes

The C++ classses discussed in this paper are available from

```
$SBL_DIR/Core/Wang_Landau/include/SBL/CSB/
```